Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Pavlin Gregor Poličar

Pitagorejska drevesa za vizualizacijo klasifikacijskih in regresijskih dreves

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: prof. dr. Blaž Zupan

Ljubljana, 2016

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Klasifikacijska in regresijska drevesa so ena od osnovnih tehnik strojnega učenja. Pri njihovi interpretaciji nam pomaga vizualizacija, ki v modernih orodjih drevo prikaže kot graf. Že pri nekoliko večjih modelih ti izrisi postanejo nepregledni. V nalogi ta problem rešite z drugačno predstavitvijo dreves, ki za izris uporabi Pitagorejska drevesa. Ta izris uporabite tudi za prikaz napovednih gozdov. Izrise implementirajte v okolju Orange in njihovo uporabnost preverite na izbranih naborih podatkov.

Zahvaljujem se mentorju, prof. dr. Blažu Zupanu, vsem v laboratoriju za bioinformatiko na FRI, prijateljem in družini.

Kazalo

Povzetek

Naslov: Pitagorejska drevesa za vizualizacijo klasifikacijskih in regresijskih dreves

Avtor: Pavlin Gregor Poličar

Program Orange je orodje za podatkovno rudarjenje, ki je za vizualizacijo klasifikacijskih in regresijskih dreves do sedaj uporabljalo klasični način predstavitve z grafom z vozlišči in povezavami. Ta predstavitev je primerna le za manjša drevesa, pri večjih drevesih pa vizualizacije z grafom zaradi prostorskih zahtev postanejo neuporabne, zato smo v diplomski nalogi preučili uporabo drugačne vizualizacije dreves. Izdelali smo interaktivni gradnik za vizualizacijo s pitagorejskimi drevesi, ki na pregleden način ponazori strukturo tako pri majhnih, kot pri velikih drevesih. Izdelali smo tudi gradnik za prikaz naključnih gozdov s pitagorejskimi drevesi, česar Orange in podobni programi do sedaj še niso podpirali. Vizualizacija je interaktivna, saj je v njej moč izbrati posamezne dele dreves in tako tudi vse pripadajoče primere. To nam omogoča hitrejše razumevanje podatkov in napovednih modelov. Vizualizacija gozdov je še posebej vizualno atraktivna in lepo ponazori kateregakoli od modelov, ki jih ta napovedna metoda strojnega učenja gradi.

Ključne besede: pitagorejska drevesa, pitagorejski gozdovi, vizualizacija dreves, Orange.

Abstract

Title: Pythagorean trees for visualizing trees in Orange

Avtor: Pavlin Gregor Poličar

Orange is a data mining toolkit that has, up to this point, only supported the classic graph method using nodes and edges for visualizing classification and regression trees. Due to space requirements this method is only useful when dealing with smaller trees. Alternative methods for visualizing trees are therefore needed. We have implemented an interactive widget that uses Pythagorean trees which conveys the tree structure clearly with small as well as with large trees. We also implemented a widget to visualize random forest using Pythagorean trees, something that Orange and similar programs did not yet support. The visualization enables users to inspect sections of trees in detail along with the selection of corresponding data in interesting branches. This could lead to better understanding of the data and underlying model. The random forest visualization is visually appealing and nicely shows each of the tree models that this popular method method infers from the data.

Keywords: pythagorean trees, pythagorean forest, tree visualization, Orange.

Poglavje 1

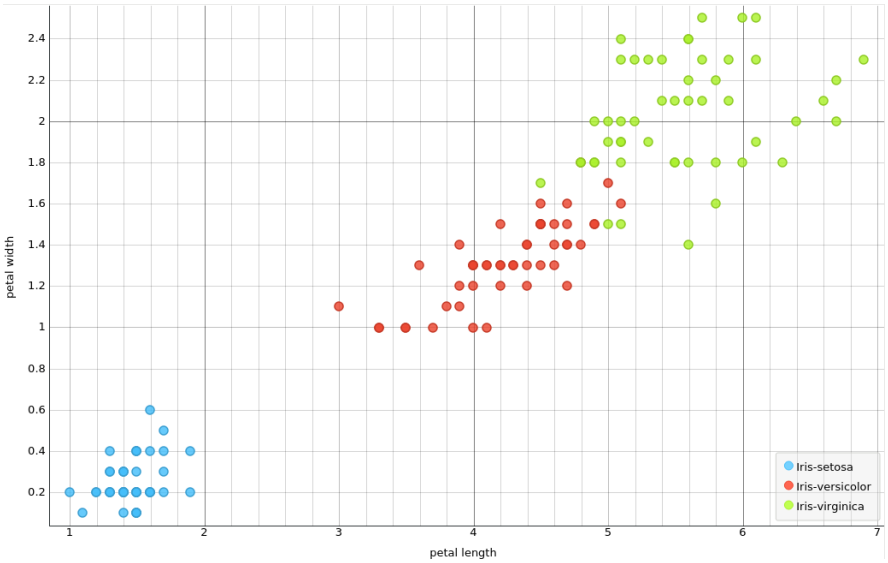
Uvod

Drevesa so ena najpogostejših podatkovnih struktur na področju računalništva. Na področju podatkovnega rudarjenja se drevesne strukture uporabljajo za gradnjo napovednih modelov. Klasifikacijska in regresijska drevesa [1] so ena najbolj osnovnih metod strojnega učenja (slika 1.1). Njihove nadgradnje s klasifikacijskimi in regresijskimi gozdovi pa danes veljajo za ene najbolj naprednih tehnik za gradnjo napovednih modelov [2].

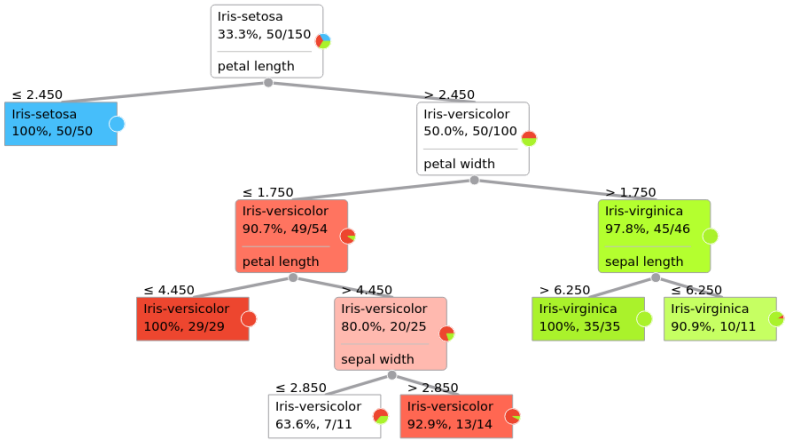
1.1 Motivacija

Ko se ukvarjamo z drevesnimi podatkovnimi modeli, si velikokrat želimo ogledati drevo, da bi bolje razumeli napovedi. Prikaz drevesa nam omogoča hiter dostop do več raznolikih informacij. Globina drevesa nam nakazuje kompleksnost modela ali opozarja na preveliko prilagajanje učni množici. Če je drevo primerno označeno, lahko hitro ugotovimo tudi, kateri so najpomembnejši atributi v vhodnih podatkih. S takšnimi informacijami lahko bolje razumemo podatke, ki jih analiziramo in iz katerih gradimo drevesa. Vizualizacija drevesa nam tudi pomaga poiskati morebitne napake, ki bi jih veliko težje opazili, če si drevesa ne bi mogli vizualno ogledati.

Čeprav poznamo več metod vizualizacij dreves, nam orodja za podatkovno rudarjenje, kot je Orange [4], tipično nudijo samo eno možnost (pri-



(a)



(b)

Slika 1.1: Razsevni diagram (a) prikazuje podatke o rožah iris [3] in vrsti posamezne rože (iris setosa, iris versicolor ali iris virginica), drevo (b) pa ustrezno klasifikacijsko drevo. Drevo pri tem klasičnem prikazu beremo od zgoraj navzdol, sprehod od korena do lista drevesa pa daje opis množice primerov v listu.

mer na sliki 3.2), ki je primerna le za manjša drevesa in postane pri večjih drevesih povsem neuporabna.

Ena izmed metod za vizualizacijo dreves so pitagorejska drevesa, tj. fraktalni pristop k risanju dreves, ki bi bil lahko primeren tudi za vizualizacijo večjih dreves. S tem pristopom lahko rešimo več problemov, s katerimi se srečujejo obstoječe rešitve, npr. prikaz večjih dreves in ugotavljanje njihovih struktur.

1.2 Cilji in struktura diplomske naloge

Cilj diplomske naloge je izdelati interaktivni gradnik v programu Orange za vizualizacijo drevesnih struktur z uporabo pitagorejskih dreves. S tem bomo uporabnikom programa Orange ponudili novo možnost za vizualizacijo dreves, ki bo pregledna tako pri majhnih, kot pri velikih drevesih.

V diplomskem delu najprej predstavimo nekatere tehnike za vizualizacijo dreves. Nato pregledamo implementacije prikazov dreves v priljubljenih orodjih za podatkovno rudarjenje. V nadaljevanju opišemo delovanje interaktivnih gradnikov, ki smo ju implementirali za orodje Orange in s pomočjo katerih v četrtem poglavju predstavimo nekaj zanimivejših vizualizacij, ki nam jih ta gradnika omogočata.

Poglavje 2

Vizualizacija dreves

Drevesa so ena najpogostejših podatkovnih struktur, s katerimi se srečujemo ne samo v računalništvu, ampak tudi v vsakdanjem življenju. Primeri dreves so družinska drevesa, organizacijska struktura podjetja, infrastruktura električne energije, datotečni sistemi in podobni.

Vizualizacija dreves (lahko jim rečemo tudi hierarhične strukture ali gnezdene strukture) je veja vizualizacije informacij, ki je namenjena prikazu drevesnih struktur in modelov [5].

Drevo je usmerjen graf $T = (V, E)$, kjer $V = \{v_1, ..., v_k\}$ označuje končno množico vozlišč in $E \subset V \times V$ končno množico povezav med vozlišči. Posebno vozlišče je koren drevesa. Koren je edino vozlišče v_r v drevesu, ki ima vhodno stopnjo $deg(v_r) = 0$. Za ostala vozlišča v_k velja, da imajo vhodno stopnjo $deg(v_k) = 1$. List je vozlišče v_k, ki ima izhodno stopnjo $deg(v_k) = 0$ [6].

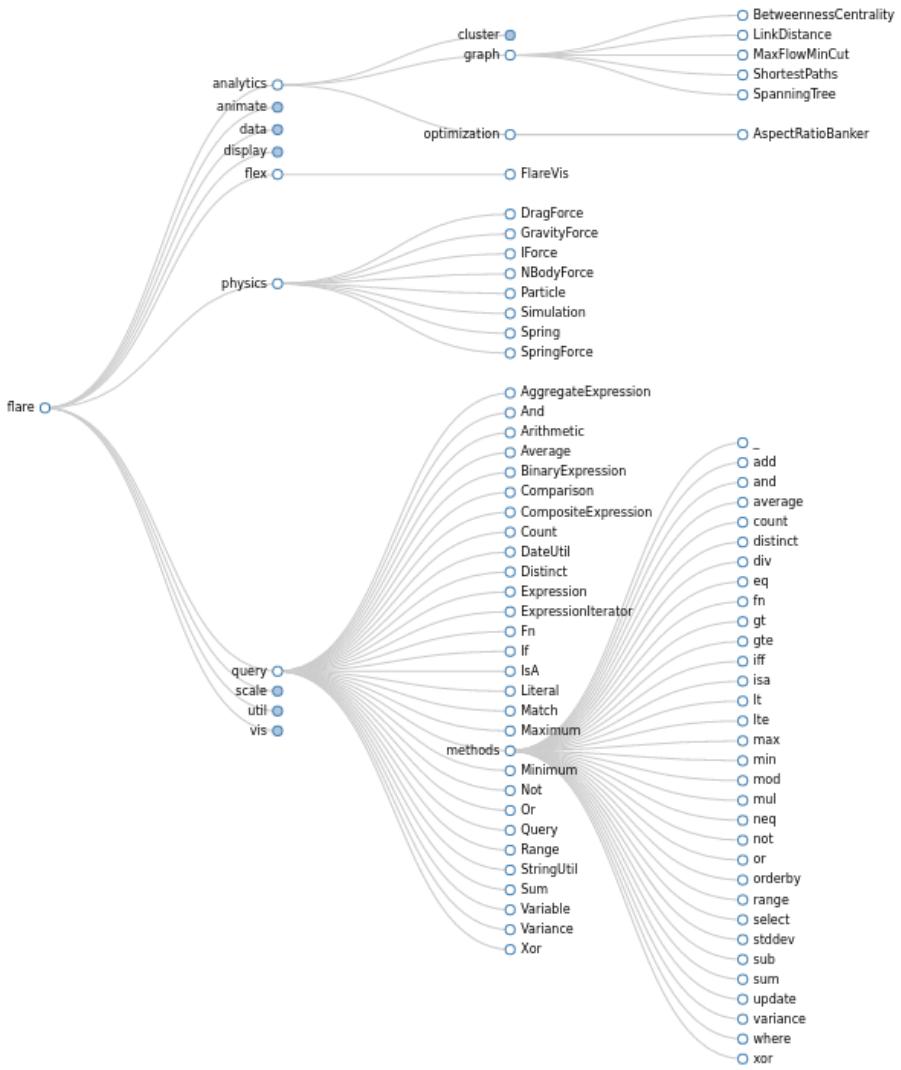
Osredotočili se bomo na drevesne strukture, ki so prisotna na področju podatkovnega rudarjenja, in sicer na regresijska in klasifikacijska drevesa.

2.1 Drevesni diagram

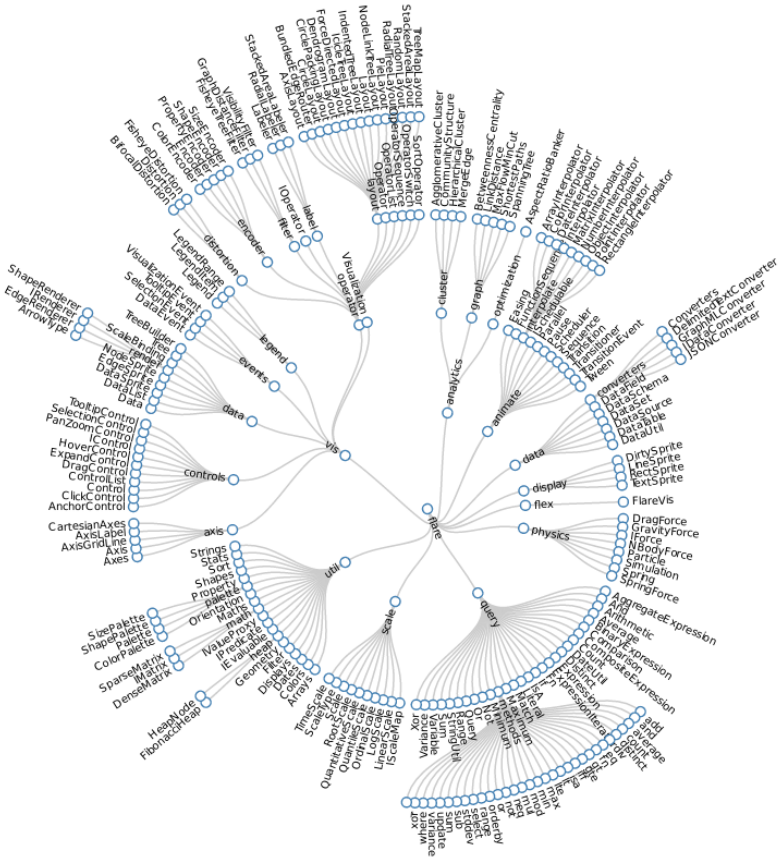
Drevesni diagrami oz. diagrami povezav (*ang. Node-link diagrams*) so verjetno najbolj naravna in intuitivna ponazoritev drevesne strukture. Prikaz drevesa s to metodo je zelo enostaven, ker ga lahko na preprost način izrišemo z vozlišči in povezavami. Velika prednost te metode je zelo jasna drevesna struktura. Tudi pri globokih drevesih struktura ostaja jasna (glej sliko 2.1).

Ta metoda se srečuje z dvema zelo pomembnima težavama, preglednim označevanjem listov drevesa in prostorsko neučinkovitostjo. Pri globokih drevesih je listov zelo veliko in jih je težko označiti na pregleden način. Spodnji del drevesa, če vzamemo pogosto obliko drevesa, kjer je koren na vrhu in listi na dnu, je zelo širok. Ker je koren samo en, bo na vrhu drevesa okoli korena veliko neizkoriščenega prostora [7].

To težavo lahko delno rešimo s krožnim diagramom povezav (slika 2.2), kjer je izkoriščenost prostora boljša, vendar lahko še vedno opazimo, da je okoli korena precej praznega prostora. Krožni diagram povezav ima tudi dodatno težavo, saj je medsebojna primerjava posameznih poddreves je otežena, ker je vsako poddrevo izrisano pod drugim naklonom.



Slika 2.1: Primer drevesa, predstavljenega z diagramom povezav. Drevo poteka od leve proti desni. Na levi strani imamo koren, na desni liste. Razvidna je glavna slabost te tehnike – neizkoriščenost prostora, saj je okoli korena veliko praznega prostora. Ob velikem številu listov so tudi oznake manj jasne in berljive (`http://bl.ocks.org/robschmuecker/7880033`).

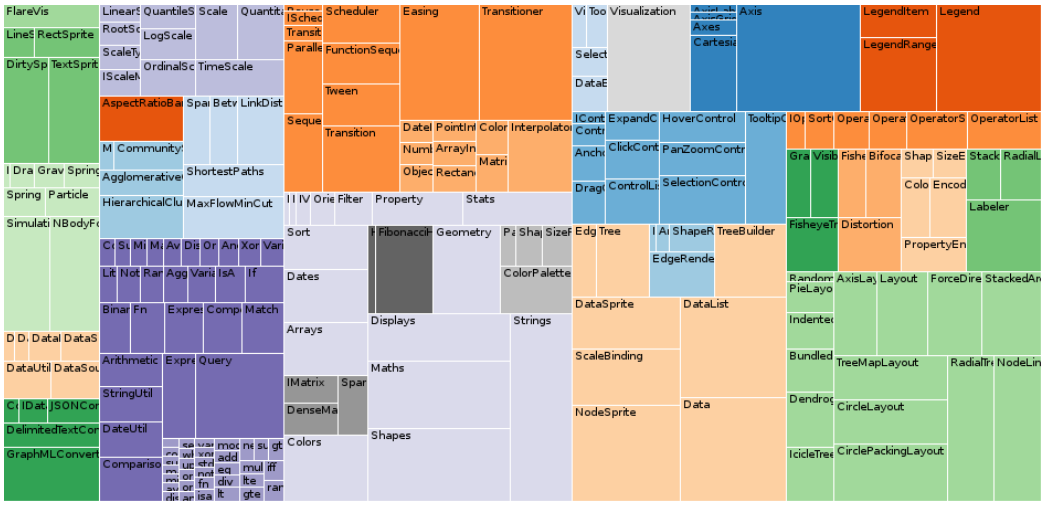


Slika 2.2: Primer drevesa, predstavljenega s krožnim diagramom povezav (`http://bl.ocks.org/mbostock/4063550`).

2.2 Ploskovni diagrami

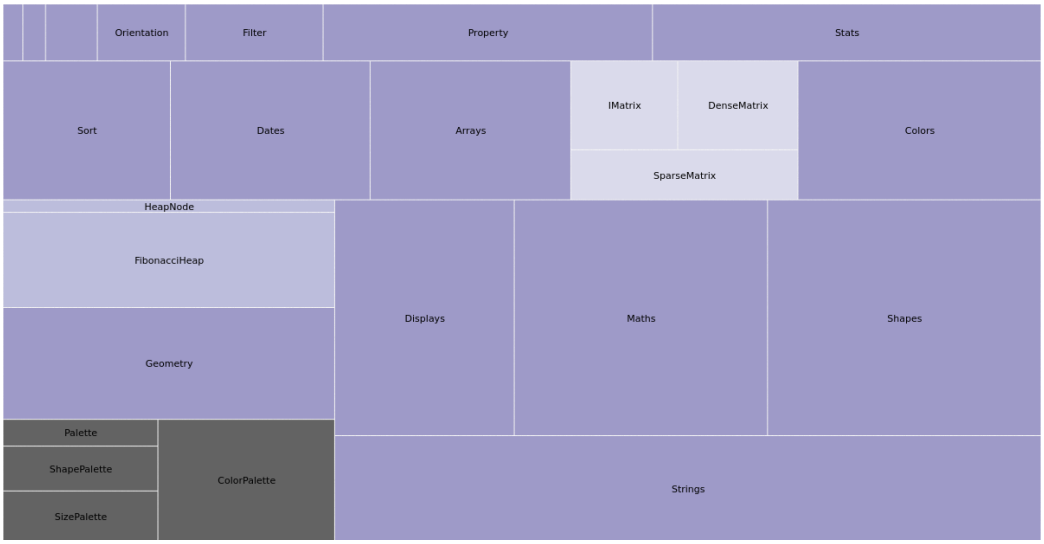
Iz želje po učinkoviti rabi prostora so se razvili ploskovni diagrami (*ang. Treemap diagrams*). Ti diagrami rešujejo ravno problem neizkoriščenosti prostora, zapolnijo namreč ves prostor, ki jim je na voljo (slika 2.3). Velika slabost ploskovnih diagramov je, da je drevesno strukturo in odnose med vozlišči veliko težje razbrati kot pri drugih metodah vizualizacij. Hierarhično strukturo lahko hitro opazimo le na najvišjih nivojih drevesa. Ti diagrami imajo dve veliki prednosti pred diagrami povezav. Prvič, nimamo težav s prostorsko učinkovitostjo, saj ne glede na globino drevesa izkoristimo celo-

tno površino, ki nam je na voljo [8]. Druga prednost ploskovnih diagramov je, da lahko z velikostjo pravokotnikov v diagramu predstavimo dodatno informacijo o drevesu [9]. Za primer vzemimo klasifikacijsko drevo, kjer bi bila velikost pravokotnika sorazmerna s številom primerov, ki spadajo v dano vozlišče.



Slika 2.3: Na ploskovnem diagramu, ki prikazuje globoko drevo, je težko razbrati končna vozlišča. Zaradi velikega števila listov so tudi tekstovne oznake slabo berljive. To lahko rešimo z interaktivnim povečevanjem diagrama (glej sliko 2.4). Struktura drevesa je slabo razvidna, hierarhično strukturo lahko opazimo samo na najvišjih nivojih drevesa. Barve označujejo razrede drevesa (`http://mbostock.github.io/d3/talk/20111018/treemap.html`).

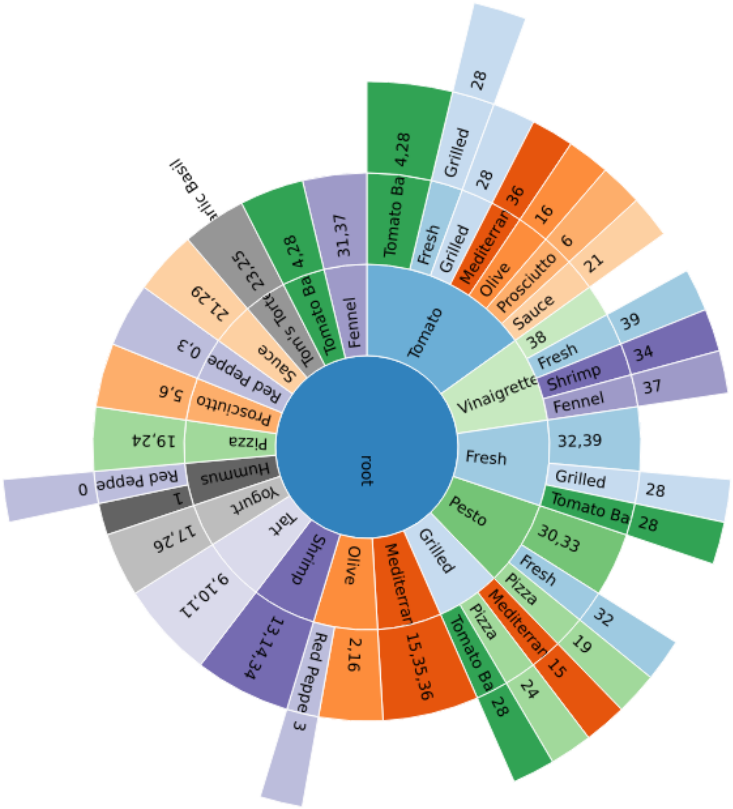
Na težave naletimo, ko imamo globoka drevesa z velikim številom listov. Takrat postane tudi pri ploskovnih diagramih pregledno označevanje listov težavno. To lahko rešimo z interaktivnostjo, saj lahko s klikom na določen odsek diagrama približamo samo izbrani del, ki potem zapolni ves prostor (slika 2.4).



Slika 2.4: Ob manjšem številu vozlišč so oznake dobro berljive. Struktura drevesa ostaja nerazumljiva (`http://mbostock.github.io/d3/talk/20111018/treemap.html`).

2.3 Sončni diagrami

Kljub prostorski učinkovitosti, ki jo nudijo ploskovni diagrami, se izkaže, da zaradi slabe razvidnosti strukture ti diagrami pogosto niso primerni za predstavitev drevesa. Sončni diagrami (*ang. Sunburst diagrams*) so krožni diagrami, ki skušajo najti kompromis med prostorsko učinkovitostjo in preglednostjo strukture [10]. Prednost te metode je boljša preglednost strukture drevesa. Njena glavna slabost je oteženo označevanje listov (slika 2.5). To težavo lahko rešimo z interaktivnim povečevanjem diagrama na izbrani odsek, podobno kot pri ploskovnih diagramih. Podobno kot pri krožnih diagramih povezav in vsemi diagrami s krožno postavitvijo je otežena tudi medsebojna primerjava poddreves, saj je vsako poddrevo izrisano pod drugačnim naklonom.



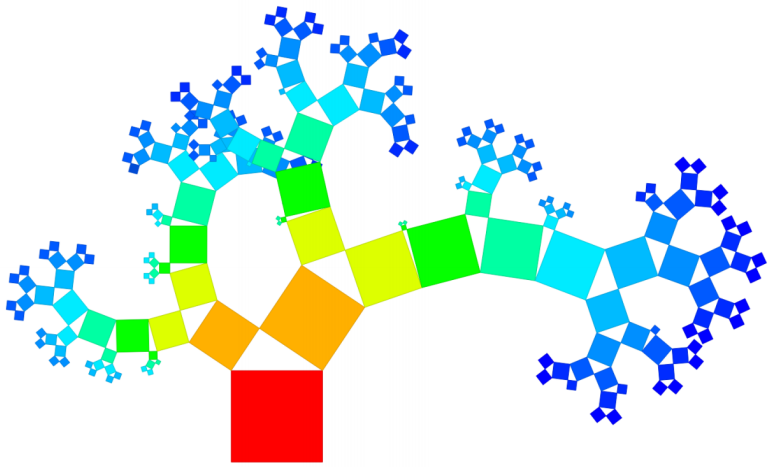
Slika 2.5: Sončni diagarm prikazuje hierarhijo jedi, ki jo je sestavil algoritem za razvrščanje v skupine. Struktura drevesa je jasna in izkoriščenost prostora je dobra (`http://codepen.io/mikefab/full/wosqD/`).

2.4 Pitagorejska drevesa

Drugačen pristop za vizualizacijo drevesnih struktur so pitagorejska drevesa. Pitagorejska drevesa *(ang. Pythagorean trees)* že več let poznamo kot fraktalni pristop za prikaz binarnih dreves (slika 2.6), vendar so pred kratkim predlagali posplošitev te metode, ki bi pristop razširila tudi na nebinarna drevesa (slika 2.7) [6]. Ta pristop se uporablja za vizualizacijo hierarhij, čeprav je vizualizacija primerna za vse drevesne strukture.

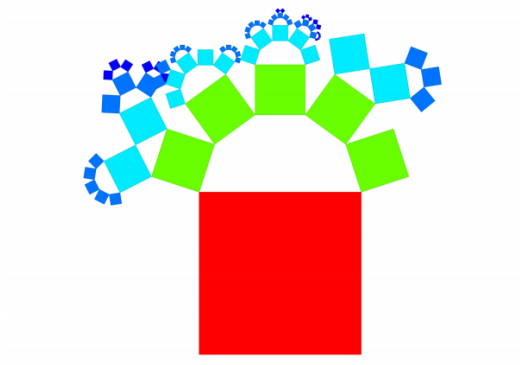
Z uporabo pristopa pitagorejskih dreves je struktura drevesa jasno razvi-

dna. Prostor je dobro izkoriščen. Glavna težava, na katero lahko naletimo, je prekrivanje vej. Na sliki 2.6 lahko vidimo primer prekrivanja, kjer ena večjih notranjih vej delno zakriva sosednjo. Temu se lahko delno izognemo z nastavitvijo različnih parametrov, s katerimi uravnamo vejitev drevesa, ter z interaktivnim izbiranjem posameznih vej drevesa. Druga težava je pregledno označevanje vozlišč. V korenu je označevanje enostavno, saj je kvadrat dovolj velik, da je tekstovna oznaka berljiva. S povečevanjem globine pa se površina kvadratov hitro manjša in pregledno označevanje vozlišč tako postane nemogoče. To težavo lahko zopet rešimo z interaktivnostjo in v primeru klasifikacijskih in regresijskih dreves, ciljni razred oz. napovedano vrednost lahko označimo z barvami.



Slika 2.6: Naključno binarno drevo, ki ga prikažemo s pitagorejskim drevesom. Barve tu označujejo globino drevesa. Na sliki lahko opazimo tudi težavo te metode – prekrivanje vej. Struktura drevesa je jasna, izkoriščenost prostora je dobra [6].

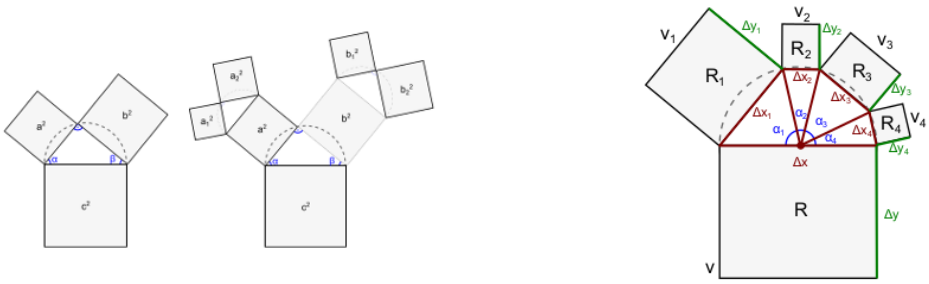
Izraz *pitagorejsko* drevo izhaja iz uporabe pitagorovega izreka $a^2+b^2 = c^2$, ki velja za pravokotne trikotnike (slika 2.8a), za risanje dreves. Pravokotni trikotniki so, v kombinaciji s kvadrati, glavni gradniki za risanje pitagorejskih dreves. Z dolžino stranice trikotnika (in s tem tudi dolžino stranice kvadrata)



Slika 2.7: Primer splošnega Pitagorejska drevesa, ki prikazuje naključno drevesno strukturo [6].

predstavimo razmerja med vozlišči, npr. število primerov, ki pripada vozlišču v klasifikacijskem drevesu.

Splošna pitagorejska drevesa razširijo osnovni pristop tako, da je moč prikazati nebinarne hierarhije ($deg_{izhodna}(v_k) \geq 2$). To dosežemo tako, da pravokotni trikotnik nadomestimo z nepravilnim konveksnim poligonom, saj je trikotnik omejen s tremi stranicami (slika 2.8b). Posplošitev predlaga tudi uporabo pravokotnikov namesto kvadratov, saj bi z nastavitvijo višine pravokotnika lahko predstavili dodatno informacijo [6].



(a) (b)

Slika 2.8: Slika (a) prikazuje postopek risanja binarnega pitagorejskega drevesa. V vsaki iteraciji na kvadrat dodamo pravokotni trikotnik, za katerega velja pitagorov izrek in na njegovi stranici pripnemo kvadrata. Slika (b) prikazuje postopek risanja splošnega pitagorejskega drevesa. Pravokotni trikotnik nadomestimo z nepravilnim konveksnim poligonom (na sliki je prikazan petkotnik). Nato ga razdelimo na enakostranične trikotnike, na katerih stranice pripnemo pravokotnike, ki jim lahko prirejamo tudi višino [6].

Poglavje 3

Vizualizacija dreves v orodjih za podatkovno rudarjenje

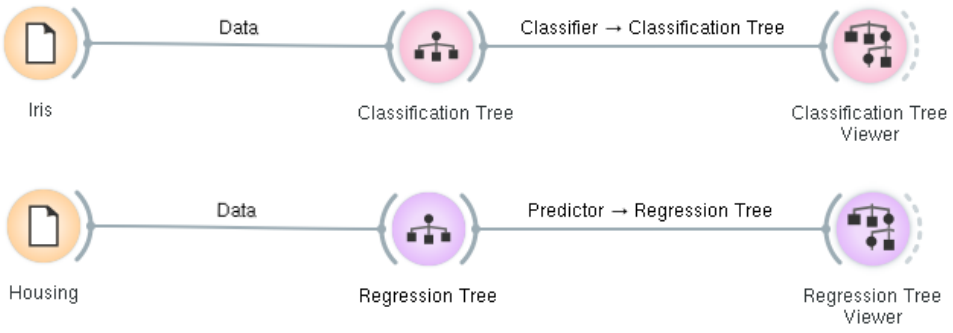
Na spletu je na voljo več odprtokodnih orodij za podatkovno rudarjenje. Ta orodja omogočajo modeliranje podatkov s pomočjo klasifikacijskih in regresijskih dreves. V tem poglavju si bomo ogledali nekaj teh orodij in skušali narediti primerjavo med njihovimi implementacijami vizualizacije dreves. Vse prikaze si bomo ogledali na zaslonu z ločljivostjo 1920×1080. Zanimalo nas bo predvsem, kako orodja prikazujejo velika drevesa, poglobili se bomo tudi v interakcijo z uporabnikom, saj je to ključen del vsake dobre vizualizacije. Naše izhodišče bo orodje Orange, za katerega smo v diplomski nalogi implementirali nov prikaz dreves.

V vseh orodjih bomo najprej sestavili majhno klasifikacijsko drevo za dobro poznan nabor podatkov *iris* [3] To drevo bomo potem prikazali in komentirali. Če se bo orodje izkazalo pri tem manjšem drevesu za uporabno, bomo orodje preizkusili še na drugem – *housing* [11], ki je regresijski problem in zahteva večje drevo. V primeru, da orodje prikaza regresijskih dreves ne podpira, si bomo večje drevo ogledali na drugem klasifikacijskem problemu, in sicer na naboru podatkov *adult* [12].

Med seboj bomo primerjali orodja Orange (v3.3.6), KNIME (v3.1.2) in RapidMiner (v7.1).

3.1 Orange

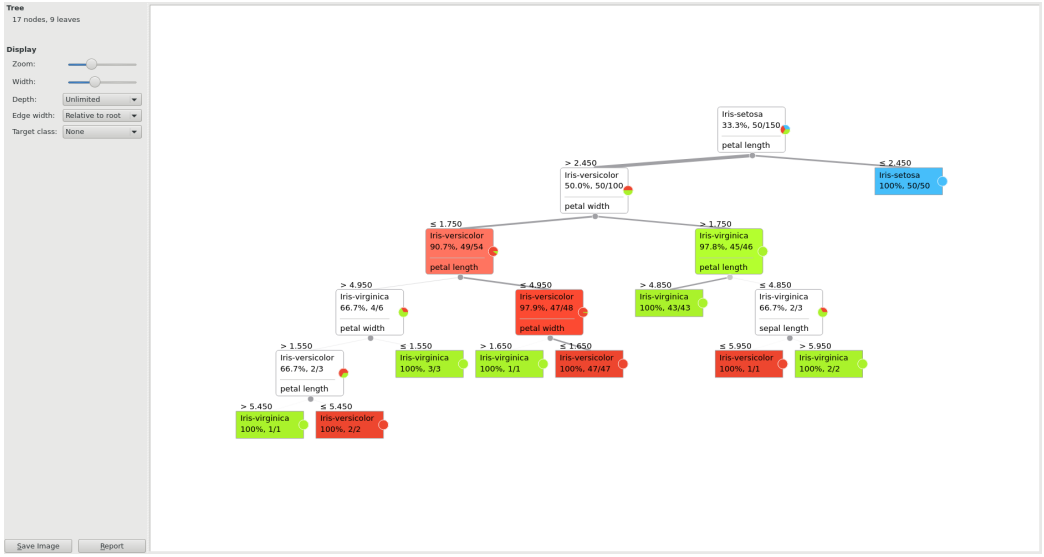
Orodje Orange [4] je program za podatkovno rudarjenje, ki se razvija na Fakulteti za računalništvo in informatiko v Ljubljani. Orange z uporabo vizualnega programiranja poenostavi kompleksnejše probleme strojnega učenja, kot je razvidno na sliki 3.1.



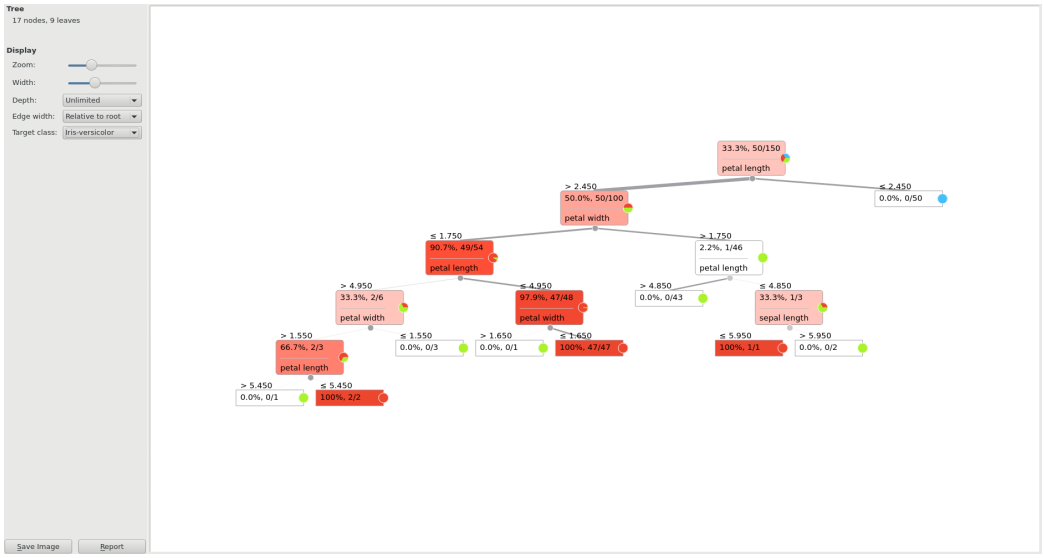
Slika 3.1: Uporabljeni delovni tok za pridobljene rezultate v orodju Orange. Delovni tok vsakega tipa drevesa je sestavljen iz treh gradnikov. Prvi gradnik naloži nabor podatkov v program, v zgornjem primeru je to podatkovni nabor *iris*, v spodnjem pa *housing*. Gradnike lahko za lažje razumevanje kompleksnejših shem preimenujemo. Drugi gradnik zgradi ustrezno drevo, ki ga nato prikaže tretji gradnik za vizualizacijo dreves.

Struktura drevesa, ki ga za nabor podatkov *iris* prikazuje slika 3.2, je zelo jasna. Vozlišča so dobro označena, takoj vidimo, kateri razred je v določenem vozlišču napovedan. Ta možnost je dodatno nadgrajena z možnostjo izbiranja ciljnega razreda, ki nam vsa vozlišča primerno obarva (glej sliko 3.3). V privzetem stanju pa manjka legenda barv. Ko ni izbran noben ciljni razred na prvi pogled ni jasno, kateri razred je v posameznih vozliščih napovedan.

Gradnik je interaktiven in nam omogoča izbiro pripadajočih primerov v poljubnem vozlišču. V primeru, da nas kakšna veja ne zanima, jo lahko skrijemo. Drevo lahko povečamo in se nato z drsniki sprehajamo po drevesu.

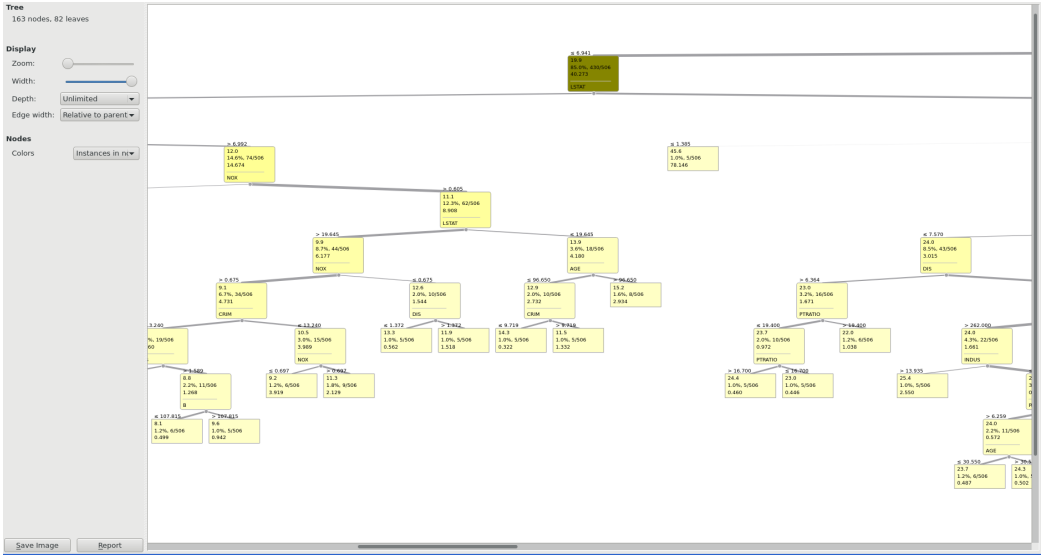


Slika 3.2: Gradnik za prikaz klasifikacijskega drevesa za nabor podatkov *iris* v programu Orange.



Slika 3.3: Če ciljni razred sami določimo v meniju na levem, nam barve prikažejo delež tistega razreda v posameznem vozlišču.

Prikaz dreves je primeren za manjša drevesa, vendar postane pri večjih drevesih neuporaben. Na sliki 3.4 je prikazan gradnik, ki prikazuje regresijsko drevo za nabor podatkov *housing*. Vizualizacija nam ne omogoča celovitega pregleda nad strukturo. Gradnik nam sicer ponudi manjšanje povečave in oženje vozlišč, vendar imamo pri obeh možnostih težave, saj nam v splošnem pri večjih drevesih ne pomagata. Pri manjšanju povečave se sorazmerno manjša tudi velikost tekstovnih oznak v vozliščih, ki postanejo pri manjših povečavah neberljive. Ponuja tudi možnost oženja vozlišč, čeprav nam oženje pogosto odreže tekstovno oznako, ki vsebuje ključne informacije. Barve nam poročajo o čistosti razbitja, kar morda za regresijsko drevo ni najbolj smiselno, saj čistost razbitja v vsakem drevesu narašča, ko se pomikamo bližje proti listom drevesa. Primernejša raba barv bi bila za regresijska drevesa napovedana vrednost v danem vozlišču, saj bi nam ta podatek večkrat prišel bolj prav kot podatek o čistosti razbitja. Barve so tudi vizualno neatraktivne. Bolje bi bilo, da bi bilo uporabniku prepuščena izbira barv z gradnikom *Color*.



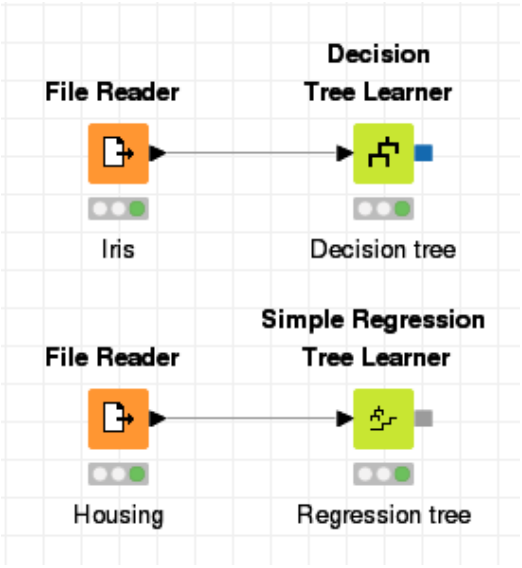
Slika 3.4: Gradnik za prikaz regresijskega drevesa za nabor podatkov *housing* v orodju Orange. Večino drevesa ni vidnega na zaslonu, kar nam onemogoči celovit pregled nad njegovo strukturo.

3.2 KNIME

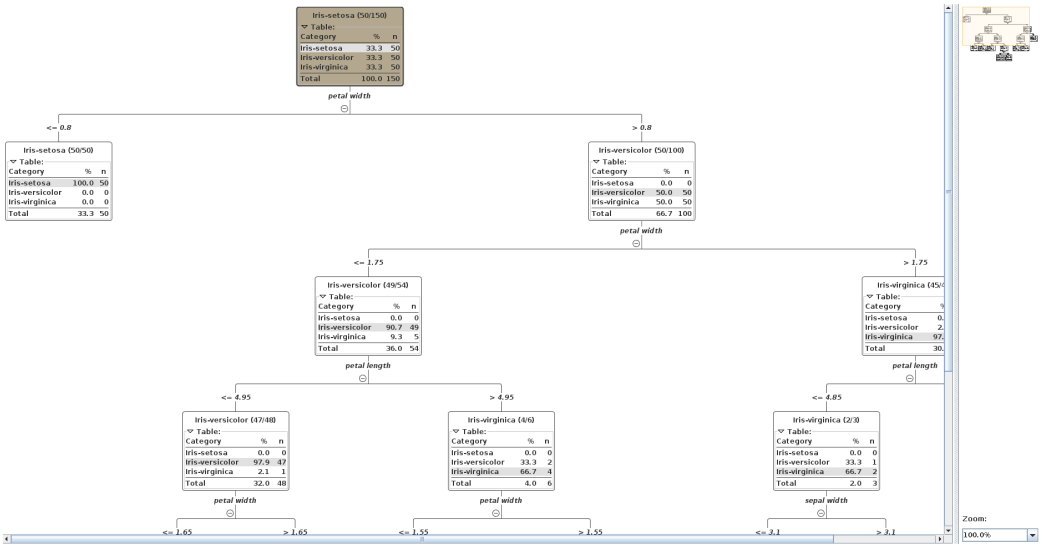
Orodje KNIME [13], podobno kot program Orange, uporablja princip grafičnega programiranja, kjer imamo na voljo več gradnikov, ki jih povežemo skupaj (slika 3.5). Orodje ima podoben prikaz klasifikacijskih dreves kot program Orange in se srečuje s povsem enakimi težavami.

Iz slike 3.6 vidimo, da je struktura drevesa za podatke *iris* jasna, vendar ima težave s prostorom, saj je že to majhno drevo preveliko, da bi ga na našem zaslonu prikazali v celoti. V tem prikazu so vozlišča dobro označena in vsebujejo vse relevantne podatke. Pomanjkljivost, ki bi jo izpostavili, je slaba označenost razreda, ki je v vsakem vozlišču napovedan. Oznake so tekstovne in ciljni razred je odebeljen, a če drevo na hitro pogledamo, ne moremo ugotoviti, v katerem delu drevesa se nahajajo posamezni ciljni razredi.

Slabo preglednost strukture lahko rešimo na podoben način kot pri pro-

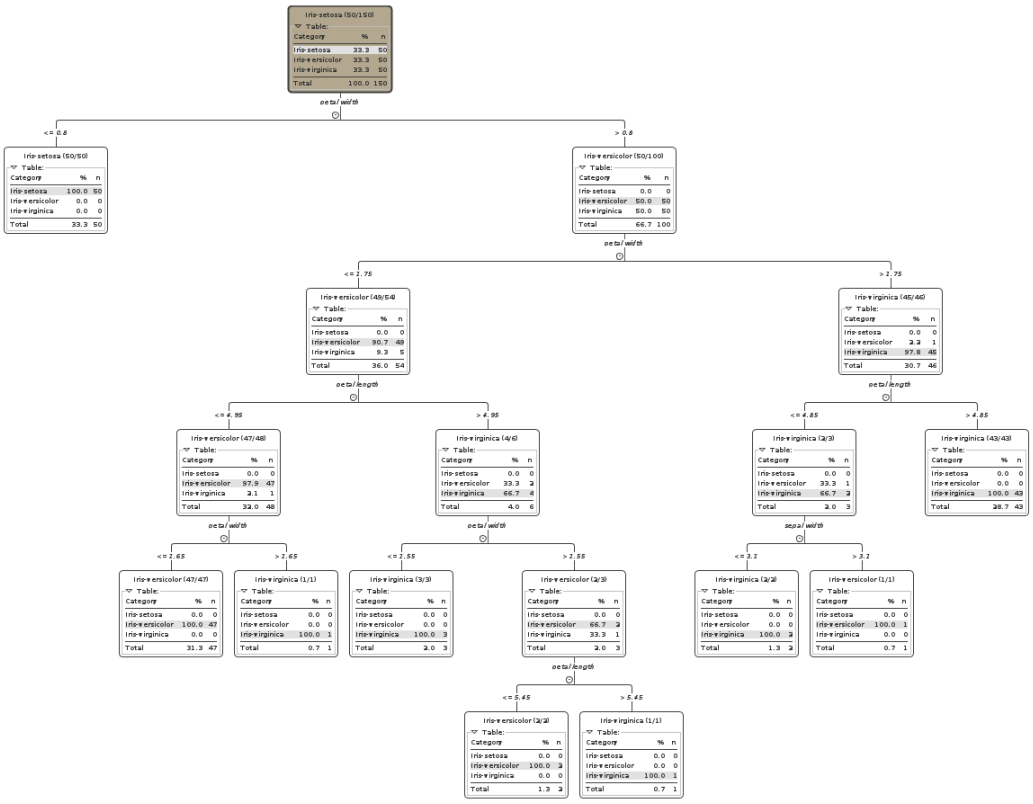


Slika 3.5: Uporabljen delovni tok za pridobljene rezultate v orodju KNIME. Orodje KNIME ima prikaz drevesa vgrajen kar v gradnik za klasifikacijska drevesa, zato lahko uporabimo samo dva gradnika.



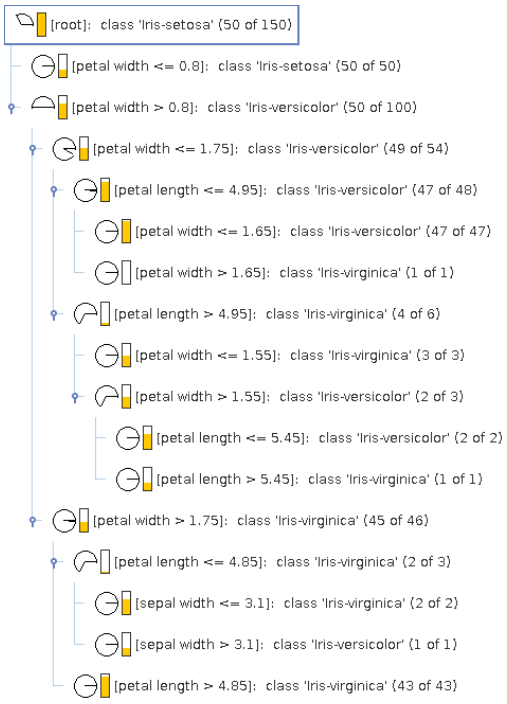
Slika 3.6: Osnovni prikaz klasifikacijskega drevesa za nabor podatkov *iris* v orodju KNIME.

gramu Orange – s povečavo, ampak tudi tu se tekstovne oznake manjšajo sorazmerno s povečavo in sčasoma postanejo povsem neberljive (glej sliko 3.7). Orodje KNIME nam ponudi tudi možnost skrivanja otrok posameznega vozlišča, kar se dobro odnese, vendar zahteva dodaten napor uporabnika.



Slika 3.7: Za ogled celotnega drevesa v orodju KNIME lahko zmanjšamo povečavo, čeprav nam to oteži oz. onemogoči branje oznak vozlišč, saj se velikost oznak manjša sorazmerno s celotno povečavo drevesa.

Orodje KNIME ponuja dodaten prikaz drevesne strukture, in sicer z metodo zamikov, kot je prikazano na sliki 3.8. Podobno kot pri osnovnem prikazu lahko tudi tu vozlišča skrčimo in razširimo, vendar se pri večjih drevesih tudi ta prikaz ne izkaže za dobrega. Tudi tu moramo porabiti kar nekaj časa, če želimo razbrati, kje se nahajajo ciljni razredi. Z metodo zamikov je tudi struktura drevesa nejasna.



Slika 3.8: Orodje KNIME ponuja tudi prikaz drevesa z metodo zamikov.

Orodje KNIME nam ne ponuja nobene interaktivnosti, s katero bi se lahko bolj poglobili v podatke. Ne omogoča izbire pripadajočih primerov posameznih vozliščih, prav tako nam ne nudi nobene dodatne možnosti za prilagajanje prikaza. Prikaza regresijskih dreves ne podpira, zato si podatkovnega nabora *housing* ne moremo ogledati. Sicer pa je iz manjšega drevesa razvidno, da tak prikaz ni uporaben za večja drevesa, zato si večjega drevesa v tem orodju ne bomo ogledali.

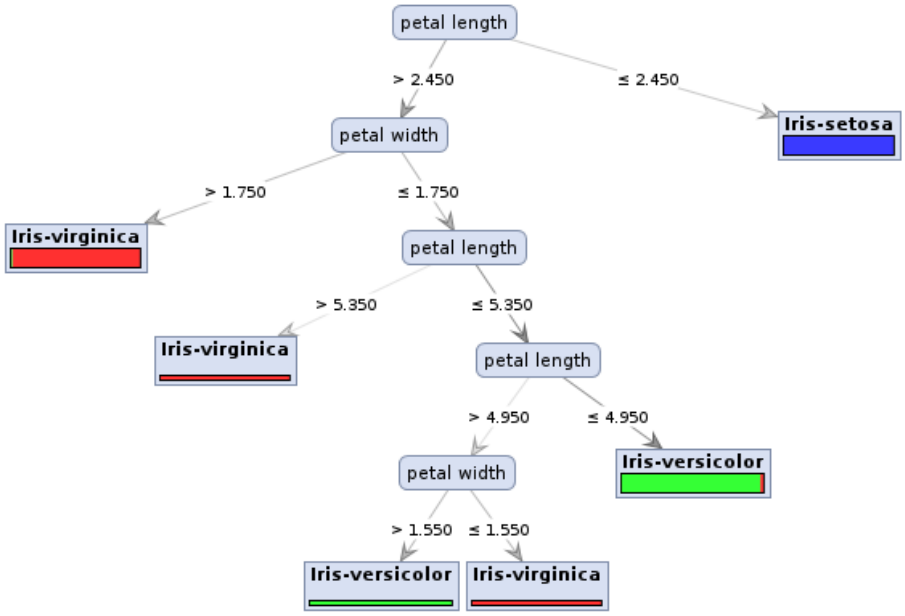
3.3 RapidMiner

Orodje RapidMiner Studio [14] uporablja, tako kot orodja Orange in KNIME, princip grafičnega programiranja, kot prikazano na sliki 3.9.

Slika 3.9: Uporabljeni delovni tok za pridobljene rezultate v orodju RapidMiner. Tako kot orodje KNIME ima tudi orodje RapidMiner vizualizacijo drevesa vgrajeno kar v gradnik za gradnjo klasifikacijskih dreves.

Orodje uporablja klasični način prikaza dreves, vendar ima za razliko od obeh omenjenih orodij malce drugačen pristop k risanju strukture. Vizualizacija namreč najbolj prostorsko učinkovite položaje posameznih vozlišč izbira dinamično, posledično je vizualizacija bolj strnjena (slika 3.10). Ker je poudarjena prostorska učinkovitost, vizualizacija prikaže zelo malo podatkov o posameznem vozlišču – o ciljnih razredih imamo podatke samo v listih. To nam sicer omogoči boljši pregled nad celotno strukturo drevesa, čeprav nam bi večkrat ravno ti podatki zelo koristili. Pogrešamo tudi podatke o številu primerov v posameznem vozlišču in o razmerjih med razredi.

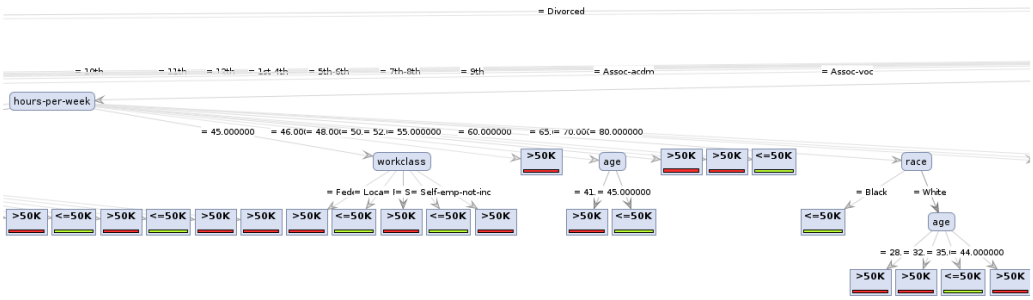
Orodje RapidMiner regresijskih dreves ne podpira, zato bomo večje drevo prikazali na naboru podatkov *adult*. Slika 3.11 potrdi težavo, s katero se srečujejo klasični prikazi dreves – prostorska učinkovitost. Kljub pametnejšim razvrščanjem vozlišč nam na zaslon uspe prikazati le majhen odsek celotnega drevesa. Vizualizacija ne podpira nikakršne interakcije z uporabnikom, ki bi to težavo lahko omilila, omogoča nam le spremembo povečave in skrivanje vseh tekstovnih oznak. Ob spremembi povečave se pomanjšajo tudi tekstovne oznake, ki so zato pri manjših povečavah povsem neberljive. Popolno skritje tekstovnih oznak nam sicer omogoči celovitejši pregled nad strukturo, toda ko to možnost onemogočimo (torej znova prikažemo teks-



Slika 3.10: Osnovni prikaz klasifikacijskega drevesa za nabor podatkov *iris* v orodju RapidMiner.

tovne oznake), se nam prikaz drevesa zamakne in moramo izbran odsek drevesa ponovno iskati. Pri večjih drevesih opazimo tudi pomanjkanje drsnika, ki bi nam olajšal preiskovanje strukture. Za pregled celotnega drevesa je potrebno veliko dodatnega dela.

Orodje RapidMiner nudi tudi tekstovni prikaz drevesa z metodo zamikov (glej sliko 3.12), ki vsebuje več podatkov kot grafični prikaz. Tudi tu pogrešamo nekaj relevantnih podatkov, kot je razmerje med posameznimi razredi v vozlišču in skupno število primerov v vozlišču. Tekstovni prikaz je tudi za večja drevesa povsem neprimeren, saj je struktura slabo vidna in sam prikaz zavzame veliko prostora.



Slika 3.11: Prikaz drevesa za podatkovni nabor *adult* v orodju RapidMiner.

Tree

```
petal length > 2.450
|   petal width > 1.750: Iris-virginica {Iris-setosa=0, Iris-versicolor=1, Iris-virginica=45}
|   petal width ≤ 1.750
|   |   petal length > 5.350: Iris-virginica {Iris-setosa=0, Iris-versicolor=0, Iris-virginica=2}
|   |   petal length ≤ 5.350
|   |   |   petal length > 4.950
|   |   |   |   petal width > 1.550: Iris-versicolor {Iris-setosa=0, Iris-versicolor=2, Iris-virginica=0}
|   |   |   |   petal width ≤ 1.550: Iris-virginica {Iris-setosa=0, Iris-versicolor=0, Iris-virginica=2}
|   |   |   petal length ≤ 4.950: Iris-versicolor {Iris-setosa=0, Iris-versicolor=47, Iris-virginica=1}
petal length ≤ 2.450: Iris-setosa {Iris-setosa=50, Iris-versicolor=0, Iris-virginica=0}
```

Slika 3.12: Tekstovni prikaz klasifikacijskega drevesa za nabor podatkov *iris* v orodju RapidMiner.

Poglavje 4

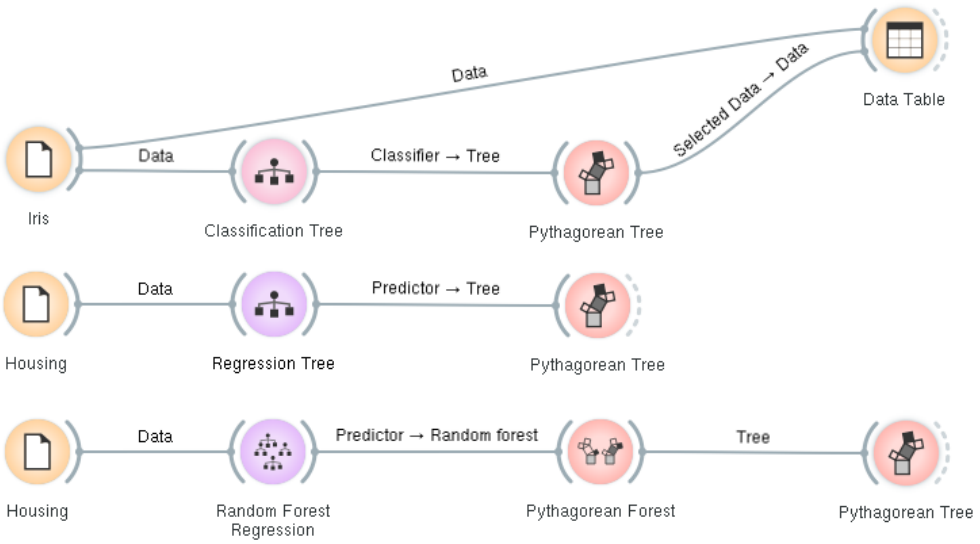
Implementacija

V diplomskem delu smo v orodju Orange implementirali dva gradnika – gradnik za prikaz klasifikacijskih in regresijskih dreves in gradnik za prikaz njunih ustreznih naključnih gozdov z uporabo pitagorejskih dreves. Gradnika sta že vključena v zadnjo verzijo orodja Orange (v3.3.7) in sta prosto dostopna na portalu Github (`https://github.com/biolab/orange3`). Gradnika sta bila že predstavljena tudi na blogu domače spletne strani orodja Orange.

4.1 Cilji implementacije

Cilj diplomskega dela je bil razviti dva gradnika – enega za prikaz klasifikacijskih in regresijskih dreves in drugega za prikaz klasifikacijskih in regresijskih naključnih gozdov. Gradnika smo želeli implementirati tako, da bi ju lahko uporabili v kombinaciji z obstoječimi gradniki, kot je prikazano na sliki 4.1.

Z implementacijo gradnikov je bilo potrebno izboljšati sistem dreves v orodju Orange, saj je bil osnovni prikaz za drevesa do sedaj razbit na dva gradnika, enega za klasifikacijska in drugega za regresijska drevesa. Z novim gradnikom smo želeli poenotiti gradnike za prikaz dreves tako, da bi za obe vrsti dreves potrebovali samo enega. Tako bi zmanjšali število gradnikov, kar pozitivno vpliva na enostavnost uporabe orodja. Podobno je bilo treba storiti za naključne gozdove, saj so bili tudi ti gradniki razbiti na dva dela. Imple-



Slika 4.1: Delovni tok za gradnika *Pythagorean tree* in *Pythagorean forest*.

mentaciji smo poenotili tako, da en sam gradnik prikaže tako klasifikacijske, kot tudi regresijske naključne gozdove.

4.2 Programska koda

Gradnik za prikaz pitagorejskih dreves smo implementirali tako, da lahko prikaže splošna (torej nebinarna) drevesa. Programska koda na sliki 4.2 vsebuje postopek za rekurzivno gradnjo pitagorejskega drevesa (celotna, neokrnjena implementacija se nahaja v repozitoriju *Github* orodja Orange v datoteki *Orange/widgets/visualize/pythagorastreeviewer.py* – vrstice 589-663).

Postopek je povsem splošen in sprejema katerokoli drevesno strukturo, ki implementira določen vmesnik. Postopek nato zgradi drevesno strukturo z vsemi potrebnimi podatki o vizualizaciji, toda drevesa ne izriše sam. Risanje prepusti drugemu delu programske kode, ki je specifičen za grafično knjižnico Qt, ki jo Orange trenutno uporablja. Morebiten prehod na drugo grafično knjižnico tako ne bi povzročal nobenih težav s samim postopkom računanja

```
def pythagoras_tree(self, tree, node, square):
    # make sure to clear out any old slopes if we are drawing a new tree
    if node == tree.root:
        self._slopes.clear()

    children = tuple(
        self._compute_child(tree, square, child)
        for child in tree.children(node)
    )
    # make sure to pass a reference to parent to each child
    obj = TreeNode(node, square, tree.parent(node), children)
    # mutate the existing data stored in the created tree node
    for c in children:
        c.parent = obj
    return obj

def _compute_child(self, tree, parent_square, node):
    weight = tree.weight(node)
    # the angle of the child from its parent
    alpha = weight * pi
    # the child side length
    length = parent_square.length * sin(alpha / 2)
    # the sum of the previous anlges
    prev_angles = sum(self._slopes[parent_square])

    center = self._compute_center(
        parent_square, length, alpha, prev_angles
    )
    # the angle of the square is dependent on the parent, the current
    # angle and the previous angles. Subtract PI/2 so it starts drawing at
    # 0 rads
    angle = parent_square.angle - pi / 2 + prev_angles + alpha / 2
    square = Square(center, length, angle)

    self._slopes[parent_square].append(alpha)

    return self.pythagoras_tree(tree, node, square)
```

Slika 4.2: Programska koda, ki vsebuje postopek za gradnjo pitagorejskega drevesa v programskem jeziku Python.

pitagorejskega drevesa. Potrebno bi bilo prirediti samo postopek za pretvarjanje strukture, ki jo prikazana programska koda proizvede v komponente nove grafične knjižnice.

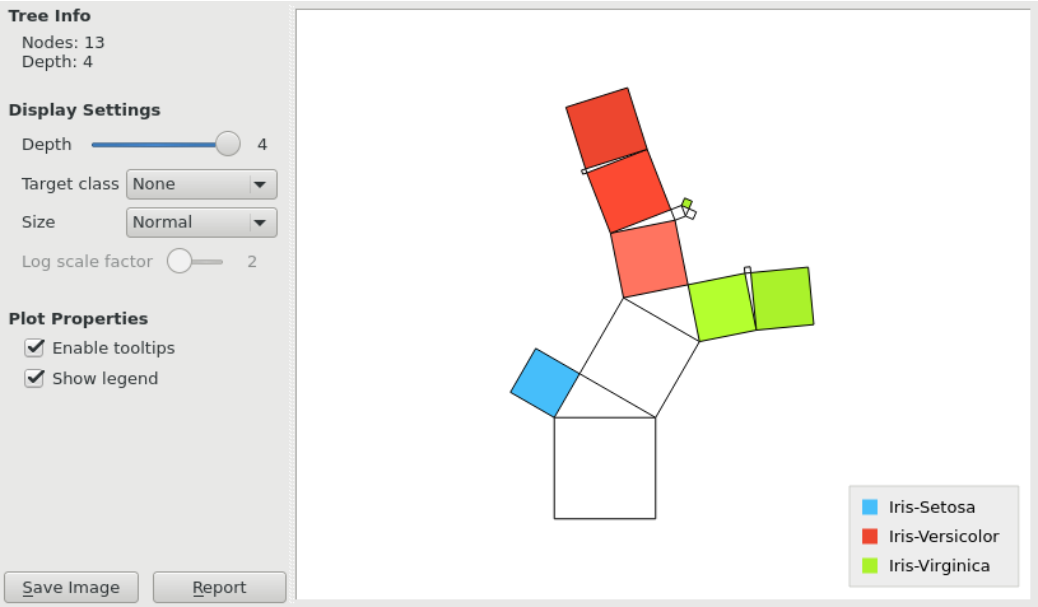
Klasifikacijska in regresijska drevesa so binarna drevesa, zato splošnost postopka ne pride do izraza in bi zato lahko ta postopek poenostavili, vendar

v primeru, da bi gradnik v prihodnosti želeli uporabiti za druga, nebinarna drevesa, ta gradnik to omogoča brez sprememb v programski kodi.

4.3 Grafični vmesnik

V tem poglavju bomo opisali delovanje novih gradnikov in možnosti, ki jih uporabniku nudijo za raziskovanje napovednih modelov.

4.3.1 Pythagorean tree



Slika 4.3: Gradnik za prikaz pitagorejskega drevesa v orodju Orange. Prikazano je klasifikacijsko drevo za nabor podatkov *iris*.

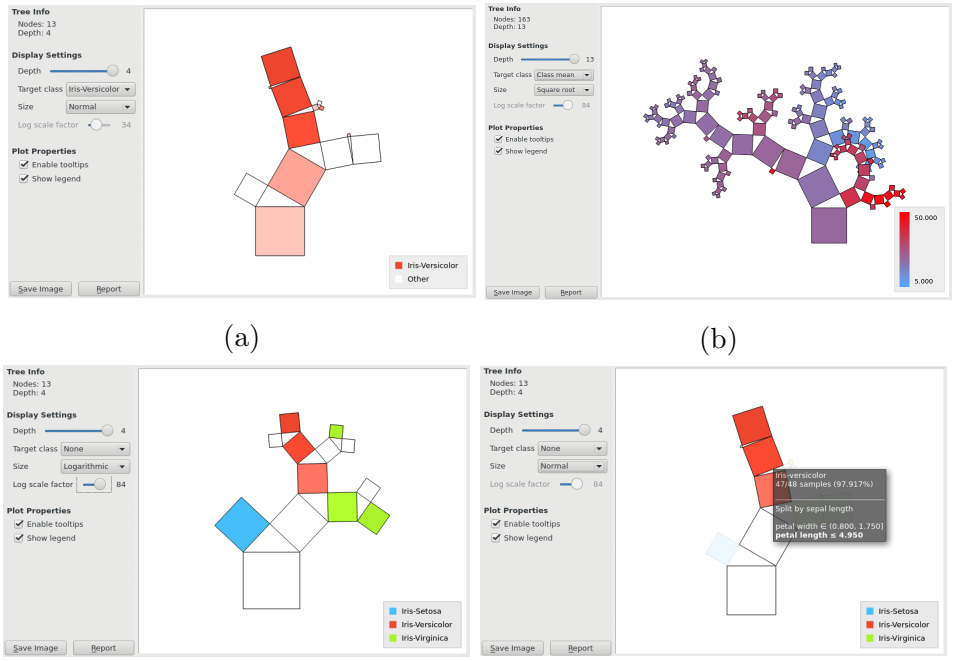
Gradnik za pitagorejska drevesa (slika 4.3) je sestavljen iz dveh glavnih delov – plošča z nastavitvami in risalno površino (tako so definirani vsi kompleksnejši gradniki znotraj orodja Orange). Gradnik na vhodnem kanalu sprejme drevesno strukturo, tj. klasifikacijsko oz. regresijsko drevo. Na izhodni kanal postavi podatke, ki pripadajo izbranemu vozlišču (če ni izbrano

nobeno vozlišče, na izhod ne postavi ničesar). Plošča z nastavitvami nam ponuja nekaj podatkov o drevesu in več možnosti za prilagajanje vizualizacije:

- **Tree info** vsebuje podatke o številu vozlišč in globini drevesa.
- **Display settings** vsebuje nastavitve, s katerimi lahko direktno vplivamo na izgled drevesa.
  - Drsnik **depth** omogoča omejitve globine v drevesu. V privzetem pogledu je prikazano celotno drevo.
  - **Target class** nam omogoča izbiro ciljnega razreda, ki ga želimo videti na drevesu. Ob izbiri ciljnega razreda se vozlišča primerno obarvajo (glej sliko 4.4a). Pri regresijskih drevesih barvanje vozlišč po razredih ne bi imelo smisla, temveč uporabniku ponudimo bolj primerne možnosti. Vozlišča lahko pustimo neobarvana, obarvamo jih lahko po napovedani vrednosti (slika 4.4b), lahko pa tudi po standardnem odklonu primerov znotraj vozlišča.
  - **Size** je nastavitev, s katero lahko prilagajamo uteži v drevesu. To nam pride prav, ko so nekatere veje vizualno zelo majhne in bi si jih želeli ogledati od bližje. Na voljo so nam uteži brez prilagajanja ter prilagajanje s kvadratnim korenom in nazadnje z logaritmom[1]. S tem je povezan tudi drsnik **log scale factor**, s katerim uravnavamo stopnjo logaritemske prilagoditve (glej sliki 4.4c in 5.2).

[1] Prilagoditev uteži najlažje predstavimo s primerom. Denimo, da ima vozlišče dva sinova z utežmi v razmerju 64:1. Brez prilagoditve uteži bo kvadrat, ki predstavlja prvega sina, veliko večji od drugega. Če uteži prilagodimo s kvadratnim korenom, pridemo do novega razmerja $\sqrt{64} : \sqrt{1} = 8 : 1$. Na podoben način na uteži vpliva logaritemska prilagoditev, z eno spremembo – $log_2(1) = 0$, kar pomeni, da eno vozlišče sploh ne bi bilo predstavljeno, zato to enostavno izpustimo. Pri našem primeru tako logaritemska prilagoditev uteži nastavi na $log_2(64) : 1 = 6 : 1$.

- **Plot properties**
  - **Enable tooltips** nam omogoča, da skrijemo pojavna okna. Ti se pojavijo, če se z miško zadržimo nad vozliščem v drevesu. Ta pojavna okna vsebujejo koristne podatke o vozlišču (slika 4.4d).
  - **Show legend** nam omogoča, da legendo, ki označuje barve vozlišč, skrijemo.
- **Save image** je možnost, ki je prisotna pri vseh vizualnih gradnikih v orodju Orange. S tem gumbom lahko lokalno shranimo sliko risalne površine. Podoben je gumb **report**, ki nam sestavi poročilo o drevesu, ki vsebuje sliko drevesa in druge koristne podatke.



(a) (b)

(c) (d)

Slika 4.4: Slike (a), (c) in (d) prikazujejo klasifikacijsko drevo za nabor podatkov *iris*. Slika (b) prikazuje regresijsko drevo za nabor podatkov *housing*.

Drevo, prikazano na risalni plošči, je interaktivna komponenta, ki omogoča enostavno raziskovanje drevesa. Interaktivnost dosežemo z večimi lastnostmi:

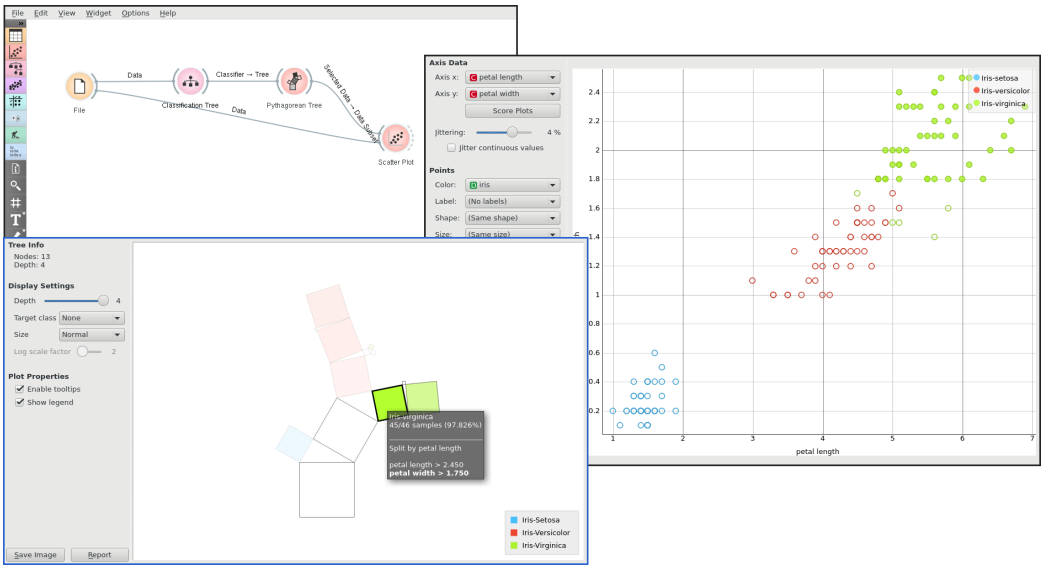
- Pri večjih drevesih večkrat pride do prekrivanja vej (prekrivanje vej lahko opazimo na sliki 4.4b) – to je težava v sami metodi pitagorejskih dreves in je neizogibna. Da nam prekrivanje ne bi otežilo razumevanja drevesa, vejo, nad katero se miška trenutno nahaja, postavimo v ospredje. To lahko delno vidimo na sliki 4.4d, kjer je označena veja v ospredju in polno obarvana, preostale veje pa so postavljene v ozadje in so delno prozorne. Tako si lahko podrobneje ogledamo tudi veje, ki so prvotno zakrite.

- Vsako vozlišče lahko izberemo in tako na izhod dobimo vozlišču pripadajoče primere iz učne množice. Tako lahko gradnik uporabimo v kombinaciji z drugimi gradniki, kar bi nam omogočilo boljše razumevanje podatkov in napovednega modela. Slika 4.5 prikazuje uporabo gradnika za pitagorejska drevesa z gradnikom *Scatterplot*. Primeri, ki pripadajo izbranemu vozlišču v klasifikacijskem drevesu, so polno obarvani na grafu gradnika *Scatterplot*. Ostali primeri, ki izbranem vozlišču ne pripadajo, imajo obarvano samo obrobo.

- Povečava drevesa je možna z miškinim koleščkom, po površini pa se lahko premikamo tako z miško, kot z drsniki. To je nepogrešljiva možnost, ko analiziramo večja drevesa.

- Pojavna okna so za prikaz koristnih informacij ključnega pomena, saj drugje na vizualizaciji nimamo prostora, da bi vozlišča označili. V pojavnem oknu imamo podatke o ciljnem razredu, čistosti razbitja, številu primerov, ki sodijo v posamezno vozlišče, atributu, po katerem razbijemo vozlišče in vseh pravilih, ki veljajo za vozlišče. Pojavna okna lahko tudi skrijemo. Velikost teksta znotraj pojavnega okna ostaja enako velika pri manjših in večjih povečavah.

- Legenda nam omogoči hitro razumevanje barv v vizualizaciji. Legendo je mogoče tudi skriti. Velikost legende ostaja enaka pri manjših in večjih povečavah.



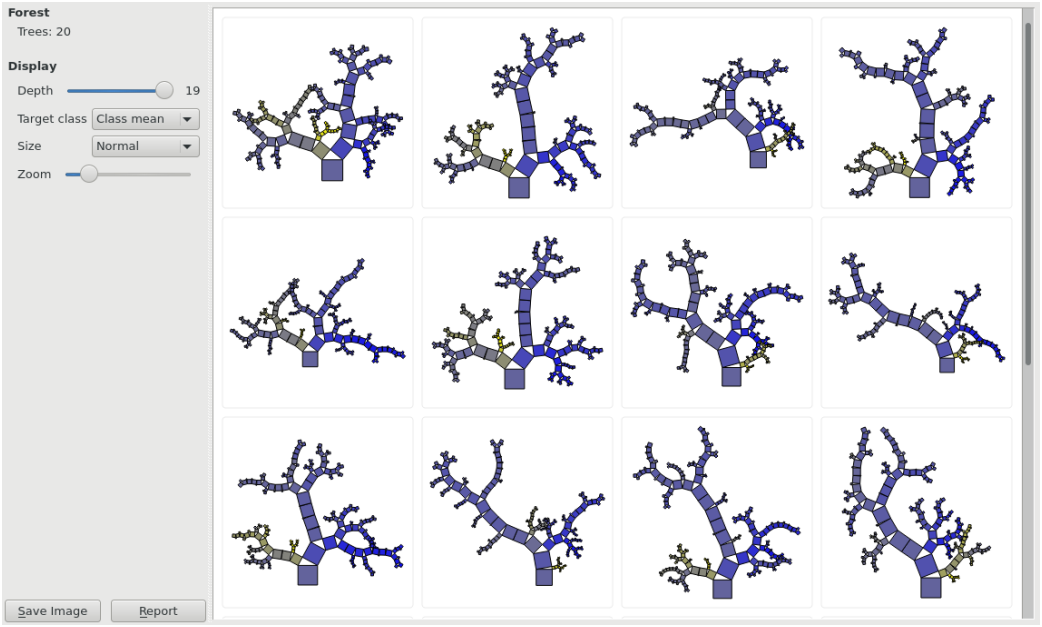
Slika 4.5: Primer uporabe gradnika za pitagorejska drevesa v kombinaciji z gradnikom *Scatterplot*. Na grafu gradnika *Scatterplot* je dobro razvidno, kateri primeri sodijo v izbrano vozlišče, saj so polno obarvani. Ostala vozlišča imajo obarvano samo obrobo. Na tak način lahko hitreje pridemo do globljega razumevanja napovednega modela, ki ga analiziramo.

4.3.2 Pythagorean forest

Tudi gradnik za pitagorejske gozdove (slike 4.6) se drži standardne postavitve gradnikov v orodju Orange in je sestavljen iz dveh delov.

Plošča z nastavitvami nam nudi podobne podatke in možnosti kot gradnik za pitagorejska drevesa.

- **Forest** vsebuje podatek o številu dreves v gozdu.
- **Display** vsebuje možnosti, s katerimi lahko direktno vplivamo na izgled prikazanih dreves. Nastavitve se prenesejo direktno na posamezno drevo in so povsem enake, kot opisano pri gradniku za pitagorejska drevesa.
  - **Zoom** je edina nastavitev, ki ni vključena v gradnik za pitagorej-

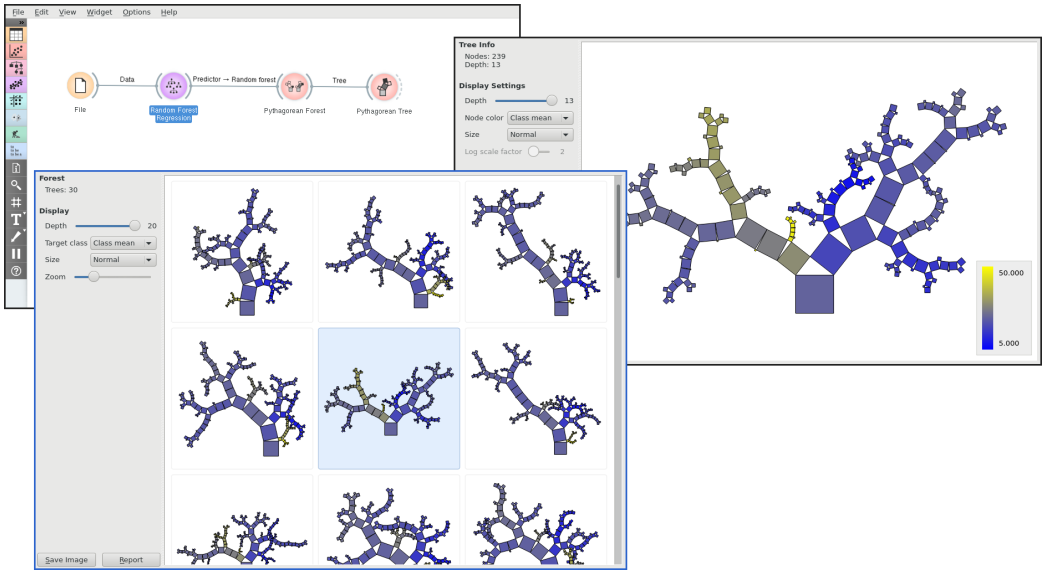


Slika 4.6: Gradnik za prikaz pitagorejskih gozdov v orodju Orange. Prikazan je regresijski gozd za nabor podatkov *housing*.

ska drevesa. S to nastavitvijo prilagajamo velikost dreves. Tako bi lahko v eno vrstico tabele vključili še več dreves, kot jih vidimo na sliko.

- Gumba **save image** in **report** sta, kot smo že omenili pri gradniku za pitagorejska drevesa, standardna za grafične gradnike v orodju Orange.

Risalna površina nam prikaže tabelo pitagorejskih dreves. Če nam kakšno drevo izgleda posebej zanimivo, ga lahko izberemo in si ga podrobneje ogledamo v drugih gradnikih za prikaz drevesnih struktur – najbolj primeren gradnik bi bil seveda gradnik za prikaz pitagorejskih dreves (slika 4.7).

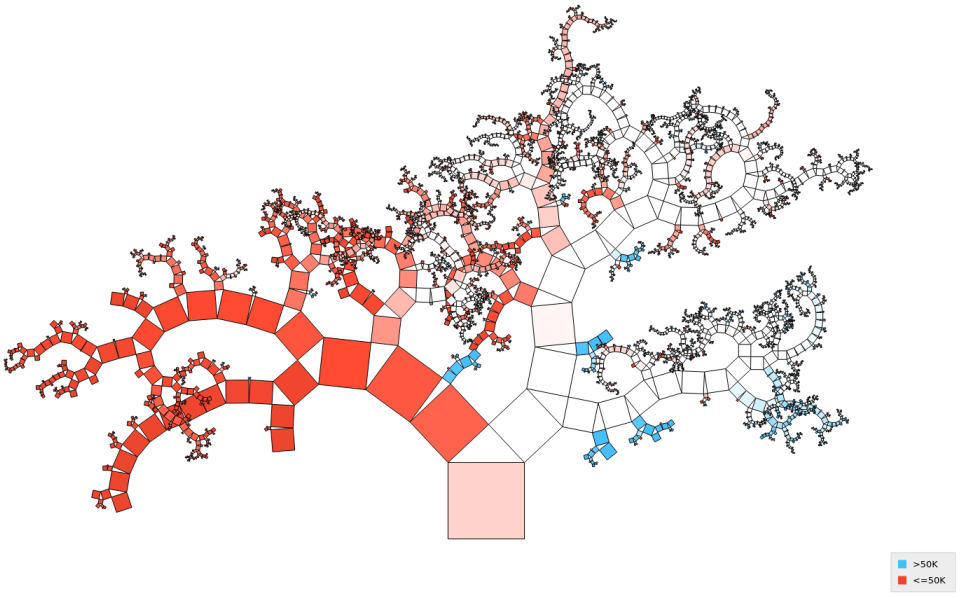


Slika 4.7: Primer uporabe gradnika za pitagorejske gozdove v kombinaciji z gradnikom za prikaz pitagorejskih dreves.

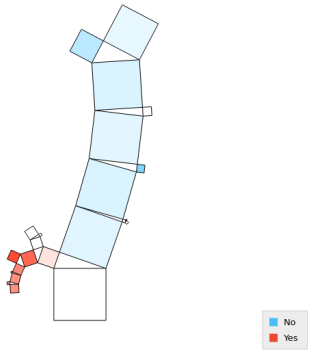
Poglavje 5

Primeri uporabe

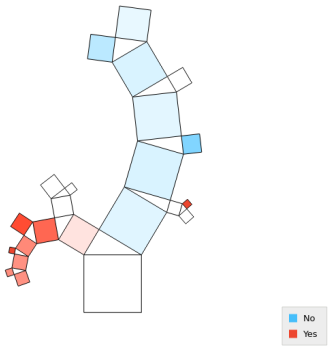
V tem poglavju bomo predstavili nekaj zanimivejših in vizualno atraktivnejših primerov pitagorejskih dreves in gozdov.



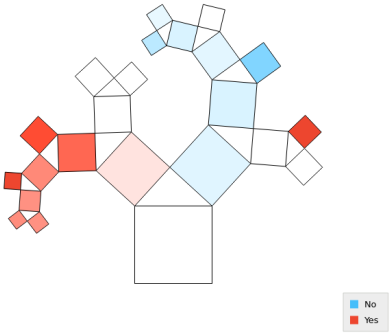
Slika 5.1: Slika prikazuje pitagorejsko drevo za nabor podatkov *adult*. Drevo vsebuje 7191 vozlišč in ima največjo globino 53. Uteži so prirejene s kvadratnim korenom.



(a) Brez prilagoditve

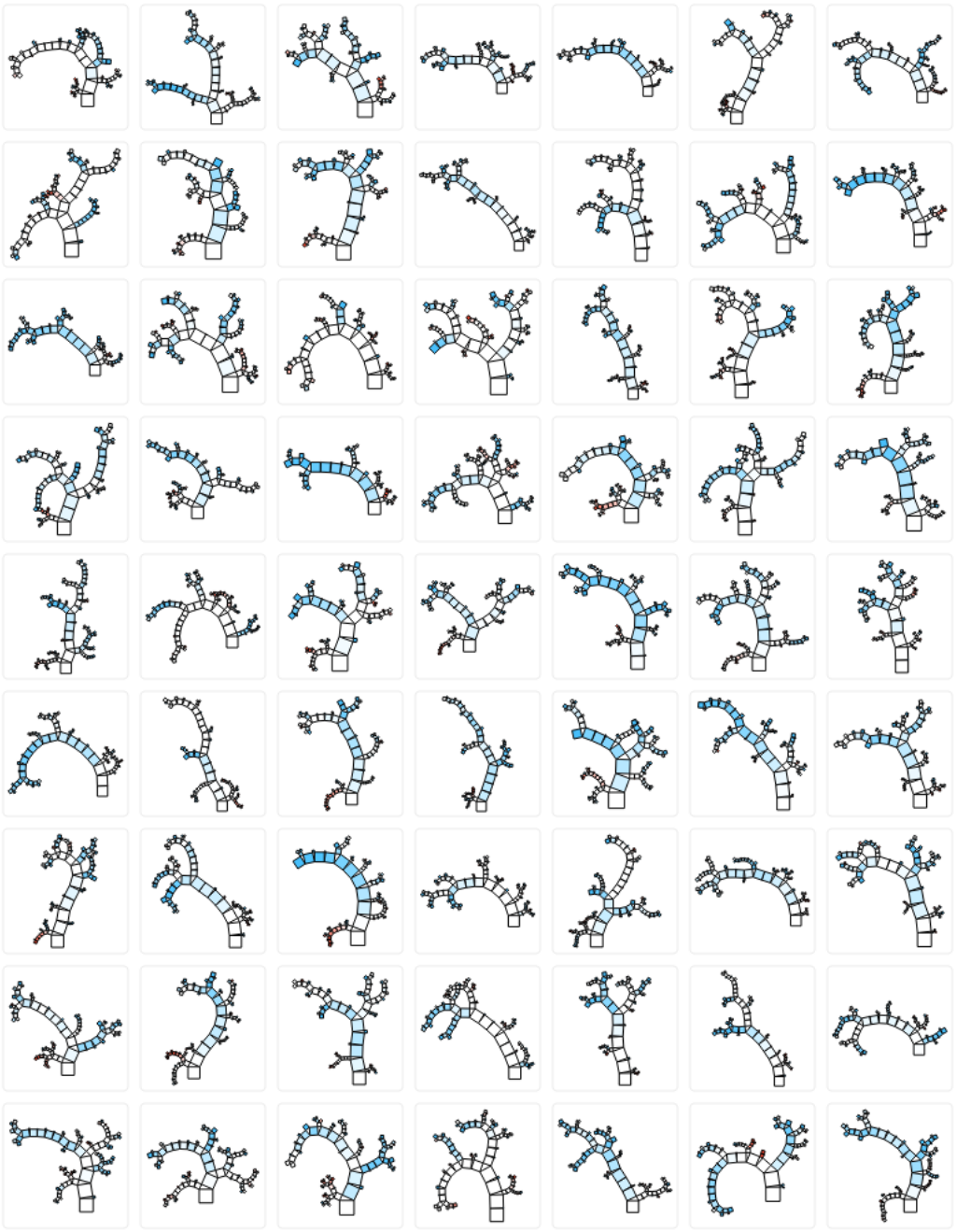


(b) Kvadratni koren



(c) Logaritem

Slika 5.2: Primerjava prilagojenih uteži na pitagorejska drevesa. Drevesa prikazujejo klasifikacijsko drevo za nabor podatkov *titanic* [15].



Slika 5.3: Slika prikazuje pitagorejski gozd za nabor podatkov *breast cancer* [16].

Poglavje 6

Sklepne ugotovitve

Z vključitvijo gradnikov v novo verzijo orodja Orange (v3.3.7) je diplomsko delo zaključeno. Vsa programska koda je prosto dostopna v repozitoriju *git* na portalu *Github* (`https://github.com/biolab/orange3`).

Orodju Orange smo dodali prikaz naključih gozdov – nekaj, česar še ni imela nobena od sorodnih odprtokodnih rešitev. Dodali smo tudi gradnik za prikaz pitagorejskih dreves, ki so prostorsko učinkoviti in dobro prikazujejo strukturo drevesa tudi pri velikih drevesih. Izboljšali smo tudi programsko implementacijo dreves in naključnih gozdov. Sedaj imamo možnost uporabe enega samega gradnika za obe vrsti dreves in naključnih gozdov, tako za klasifikacijske, kot za regresijske probleme.

Potrebno je popraviti klasični prikaz dreves v orodju Orange tako, da bo tudi ta z enim gradnikom podpiral obe vrsti dreves, saj je ta trenutno razdeljen na dva gradnika. V prihodnosti bi lahko implementirali tudi kakšen dodaten gradnik za vizualizacijo drevesnih struktur. Tukaj moramo biti pazljivi, saj filozofija orodja Orange ni taka, da uporabniku ponudimo čim več gradnikov, temveč da mu ponudimo le nekaj dobrih možnosti. Izmed analiziranih tehnik za prikaz dreves se nam zdijo najbolj zanimivi sončni diagrami, ki bi bili lahko primerni tudi za pregleden prikaz drevesnih struktur.

Literatura

[1] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[2] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems," *J. Mach. Learn. Res*, vol. 15, no. 1, pp. 3133–3181, 2014.

[3] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[4] J. Demšar, T. Curk, A. Erjavec, Črt Gorup, T. Hočevar, M. Milutinovič, M. Možina, M. Polajnar, M. Toplak, A. Starič, M. Štajdohar, L. Umek, L. Žagar, J. Žbontar, M. Žitnik, and B. Zupan, "Orange: Data mining toolbox in python," *Journal of Machine Learning Research*, vol. 14, pp. 2349–2353, 2013.

[5] H.-J. Schulz, "Treevis. net: A tree visualization reference," *Computer Graphics and Applications, IEEE*, vol. 31, no. 6, pp. 11–15, 2011.

[6] F. Beck, M. Burch, T. Munz, L. Di Silvestro, and D. Weiskopf, "Generalized pythagoras trees for visualizing hierarchies," in *Information Visualization Theory and Applications (IVAPP), 2014 International Conference on*, pp. 17–28, IEEE, 2014.

[7] C. Plaisant, J. Grosjean, and B. B. Bederson, "Spacetree: Supporting exploration in large node link tree, design evolution and empirical evalu-

ation," in *Information Visualization, 2002. INFOVIS 2002. IEEE Symposium on*, pp. 57–64, IEEE, 2002.

[8] C. Ware, *Information visualization: perception for design*. Elsevier, 2012.

[9] B. Shneiderman, "Tree visualization with tree-maps: 2-d space-filling approach," *ACM Trans. Graph.*, vol. 11, no. 1, pp. 92–99, 1992.

[10] J. Stasko and E. Zhang, "Focus+ context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations," in *Information Visualization, 2000. InfoVis 2000. IEEE Symposium on*, pp. 57–65, IEEE, 2000.

[11] D. A. Belsley, E. Kuh, and R. E. Welsch, *Regression diagnostics: Identifying influential data and sources of collinearity*. Wiley, 1980.

[12] R. Kohavi, "Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid.," in *KDD*, vol. 96, pp. 202–207, 1996.

[13] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, C. Sieb, K. Thiel, and B. Wiswedel, "KNIME: The Konstanz Information Miner," in *Studies in Classification, Data Analysis, and Knowledge Organization (GfKL 2007)*, Springer, 2007.

[14] M. Hofmann and R. Klinkenberg, *RapidMiner: Data Mining Use Cases and Business Analytics Applications*. Chapman & Hall/CRC, 2013.

[15] R. Dawson, "The "unusual episode" data revisited," *Journal of Statistics Education*, vol. 3, no. 3, pp. 1–7, 1995.

[16] W. H. Wolberg and O. L. Mangasarian, "Multisurface method of pattern separation for medical diagnosis applied to breast cytology.," *Proceedings of the national academy of sciences*, vol. 87, no. 23, pp. 9193–9196, 1990.